Autocross

Self Driving Car with communication

System Design Rev 0.1

Members:
Group 14
Nicholas Willison - 1208917
Yunan Yan -1218102
Sushant Anantharam - 1217395
James Priebe - 1135001
Patrick Jakubowski - 1144606
Abhishek Mukherjee - 1151803

0. Table of Revisions

Version	Date	Author	Description
0.1	12/18/2016	Nicholas Willison	Created document outline
0.2	12/21/2016	Group	Initial Release, Rough Release

1. Introduction

1.1 Document Purpose

The purpose of the document is to communicate and establish a viable system design for the AutoCross system. The AutoCross system is an autonomous RC car control system that will be able to successfully communicate with other autonomous cars through a standardized V2V (Vehicle to Vehicle) protocol. This V2V protocol will be used to facilitate car behaviour at an intersection, specifically when 2 cars arrive at an intersection at the same time.

1.2 System Purpose

The purpose of this project is to explore and engineer a proof of concept system autonomous RC and V2V system. The system shall adhere to the following criteria:

1. AutoCross shall operate autonomously
2. AutoCross will implement lane following on a standardize track (track implementation TBD)
3. AutoCross shall stop at an intersection
4. AutoCross shall be able to communicate with other cars through a standardized protocol
5. AutoCross shall use the V2V protocol to successfully facilitate at least 2 compatible cars through an intersection who arrived at the same time
6. AutoCross shall implement basic static obstacle avoidance
7. AutoCross shall cost no more than $400.

1.3 Goals

The following are project specifics goals for AutoCross:

1. Do not crash
2. Stay within track lanes
3. Detect a stop sign and stop at the track intersection
4. Smooth vehicle motion (proper PID tuning)

5. Avoid static obstacles by either stopping or swerving
    a. May break lane following rule
    b. For simplicity, the first version it will not check oncoming lanes
6. Efficient well managed code
7. V2V protocol will be established and standardized between all groups
8. V2V protocol shall indicate the car's current state and intention

2. Design Overview

2.1 High Level Functions

The system can be broken down into 3 high level functionalities. Functionalities 1 and 2 are marked as highest priority and as such will receive the most attention.

1. *Ability to drive autonomously including:
    a. Lane following
    b. Stopping at an intersection
2. *Ability to communicate with other autonomous cars with the intent to successfully travel through an intersection after arriving at an intersection with 1 or more other cars
3. Obstacle avoidance with lane correction

Note: Functionalities marked with * have highest priority.

2.2 System Context

The following diagram outlines the overall context of the system. AutoCross will utilize the existing motor and servo control circuits on the RC car. Utilizing these circuits AutoCross can control the speed of the car by varying PWM voltages as well as controlling the wheel angle by manipulating the steering servo. In addition, AutoCross will have an on-board camera which is will use for multiple purposes including lane detection, distance detection and object detection (stop sign and obstacles). The system will also feature an ultra-sonic sensor to aid in object detection. The last component to the system is the Vehicle to Vehicle (V2V) module will broadcast and receive state and other associated information to surrounding RC cars.

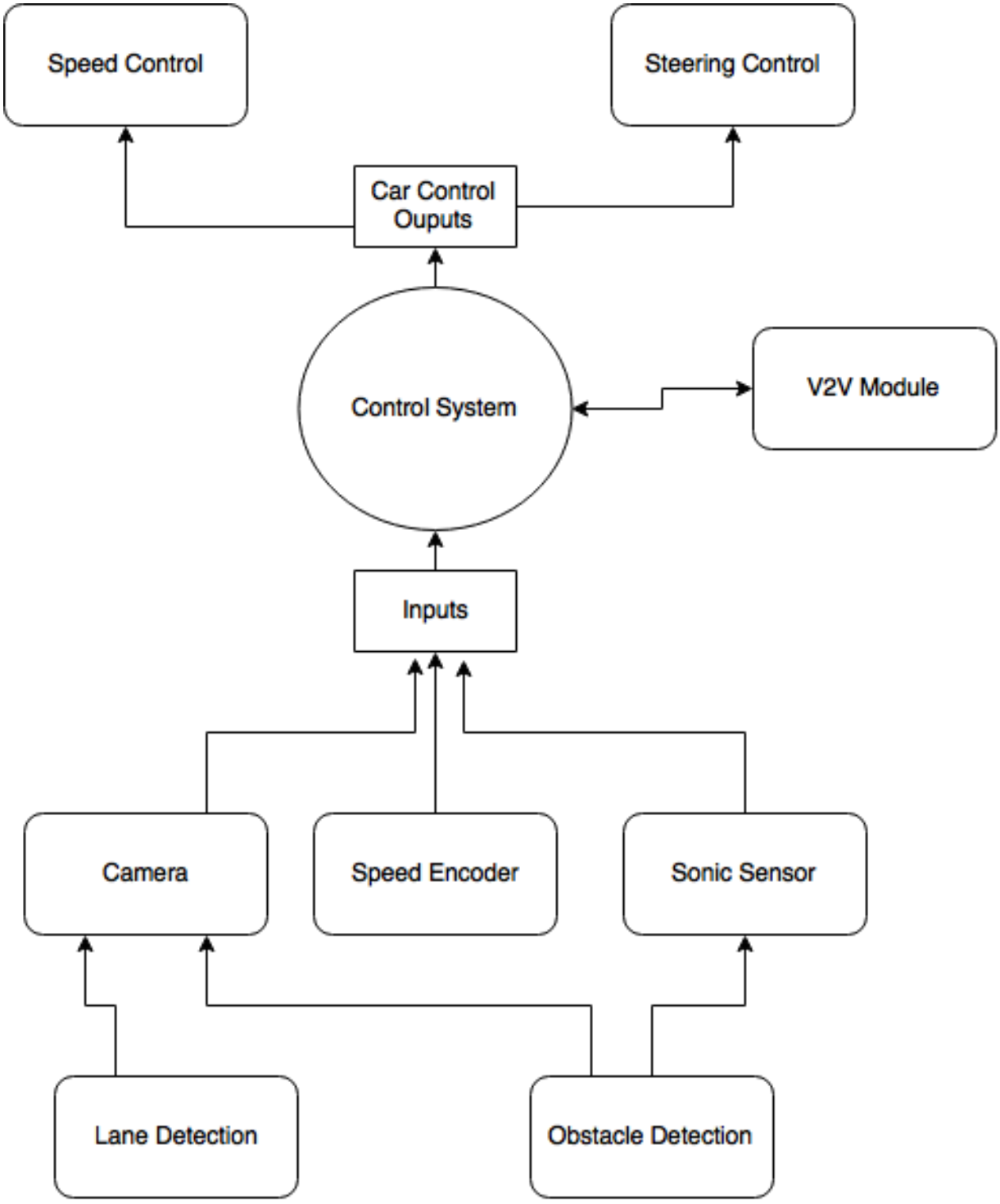


Figure 1: System Context Diagram

2.3 System Overview

The following is a system overview diagram of AutoCross.

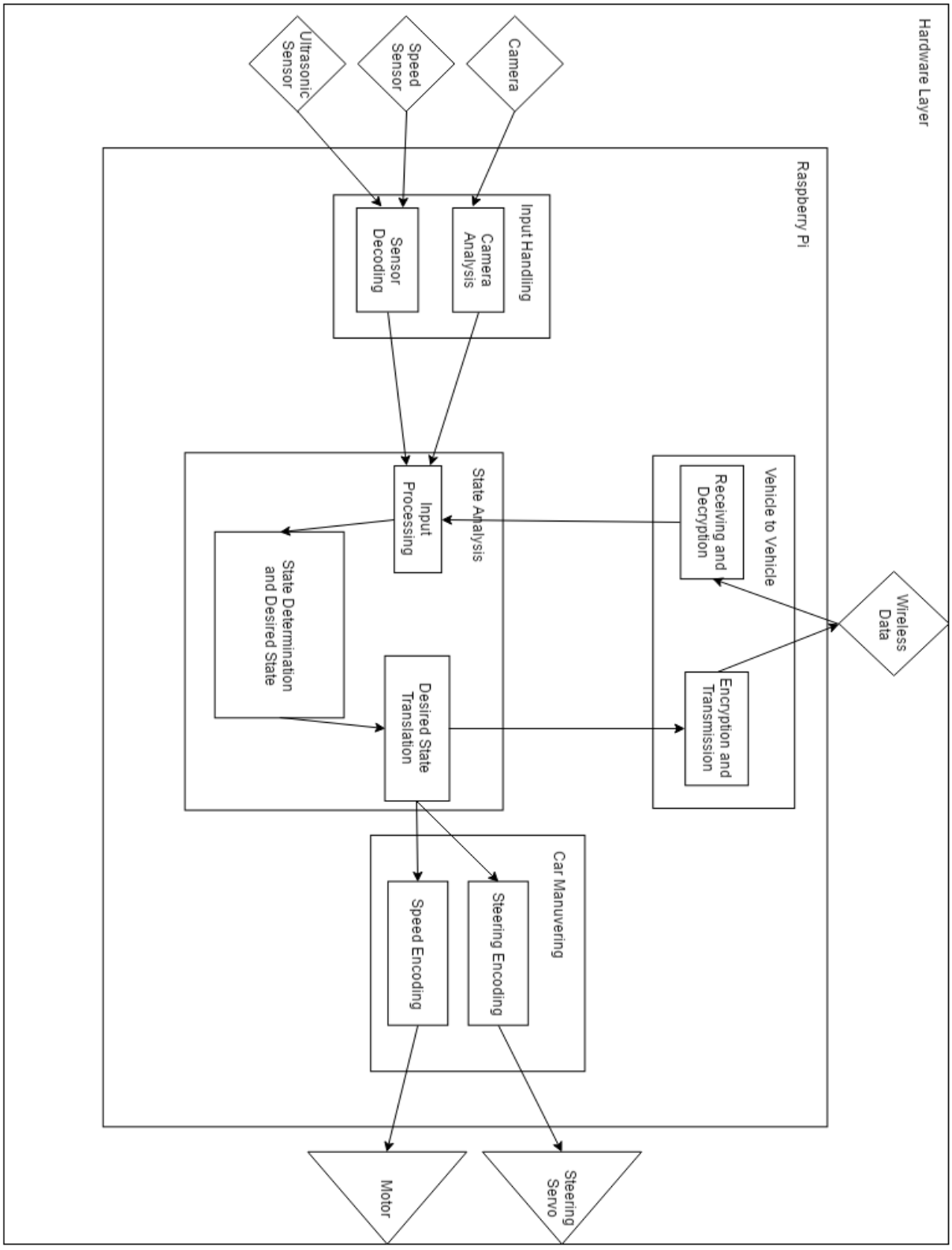


Figure 2: System Overview

2.4 Vehicle Chassis

The team choose Tamiya TT-01 as the vehicle chassis since its weight is relatively light and the position of the battery will provide a low center of gravity. It also contains the dry friction shock absorber to reduce the impact damage.

Specification:

- Length: TBD
- Width: TBD
- Height: TBD

In order to drive the vehicle safely and achieve all the goals, the following requirements need to be satisfied:

- The vehicle should always travel within the lane and the width of the vehicle cannot exceed the width of the course.
- The vehicle is able to turn all the directions (straight/left/right/back).
- The vehicle needs to move smoothly and it is able to stop within the safety distance for all the traffic rules and also for any dangerous situation happen. The brake control system is essential in this case.
- The vehicle requires power saving to save the battery.
- The vehicle needs to be made by solid and relatively light material.

2.5 System Hardware

Hardware	**Description**
RC Vehicle	Serves as the basis of the system. The vehicle includes the DC drive motor as well as the servo controlling the the left and right wheel direction.
Raspberry Pi 3	Miniature linux based computer with IO(input/output) capability. This Pi will serve as both the image processing and IO unit. The Pi will use OpenCV to process a live camera feed to detect lanes, objects and stop signs. In addition, the Pi will use its IO pins to control the DC drive motors and steering servos on the RC car. Details: ● Broadcom BCM2837 SoC ● 1.2 GHz quad-core 64-bit ARM Cortex-A52 processor ● 1GB RAM ● 40 GPIO pins

	The Raspberry Pi 3 is small yet powerful and should provide ample processing power for both IO and vision processing.
Pi Camera	The Pi camera is a module made specifically for the Raspberry Pi. The Pi Cam plugs directly into a dedicated camera port on the Pi. The camera has a small 5MP fixed focus sensor which should provide plenty resolution for image processing. Details: ● 5MP (2592 x 1944) ● Streaming: 1080p@30fps, 720p@60fps, 640x480p@90fps
RF24L01 2.4GH Wireless/Radio Transceiver	The RF24L01 is 2.4GH radio transmitter and receiver. This component will serve as the basis for the Vehicle to Vehicle (V2V).
Hall Effect Sensor	The hall effect sensor is used to count the change in a magnetic field. Placing magnets on the a RC wheel with the sensor mounted on the wheel drive shaft the angular velocity can be calculated. GIven the angular velocity and the wheel diameter the linear velocity can be determined
Ultrasonic sensor	This sensor can be used to detect and determine the distance between the sensor and other objects. The sensor works by sending and receiving bounced ultrasonic waves.

Table 1: System Hardware Description

2.6 Sensor Component Placement

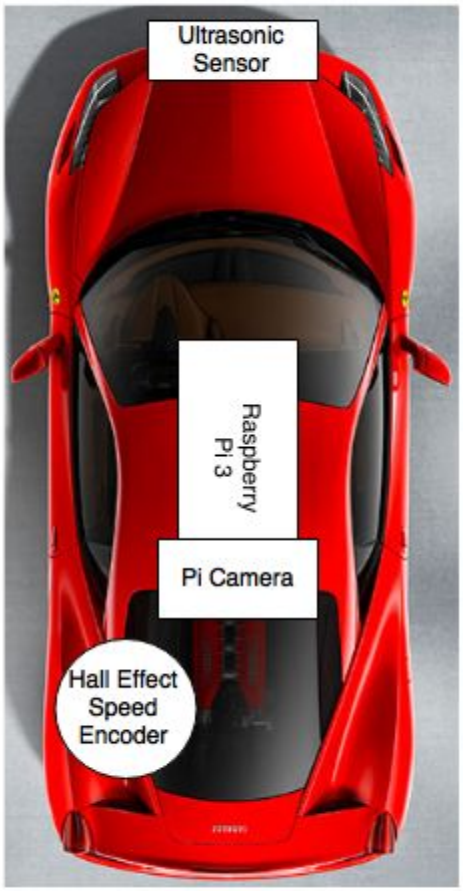


Figure 3: Sensor location

2.7 System State Diagram

The following is a general overview of a discrete time model of the system

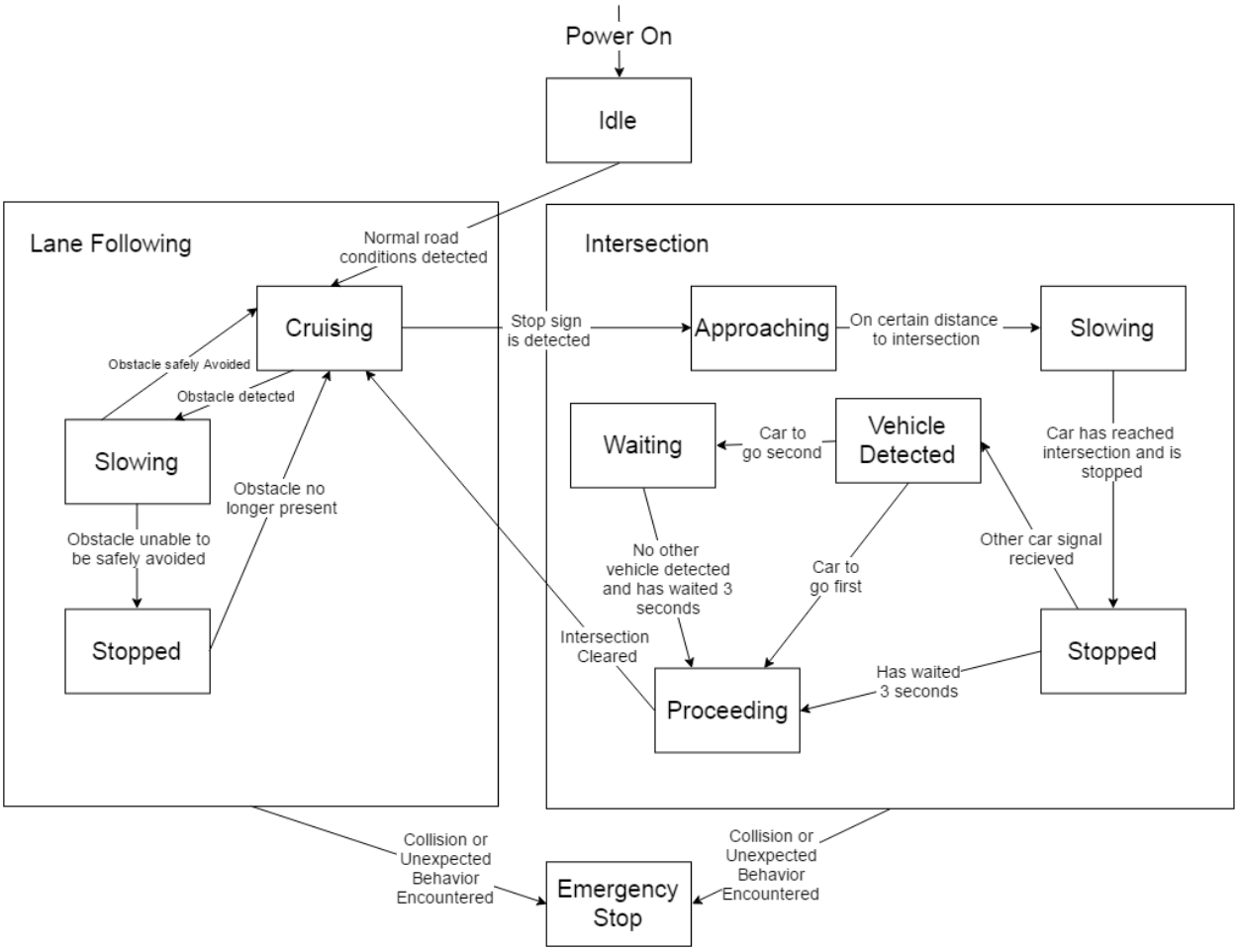


Figure 3: State Diagram

Note that it is possible for the system to enter an emergency stop from any state, this is a state where the only recovery is power off and power back on, this is a means of ensuring that the hardware does not become damaged, nor threatens to damage any other equipment. This state is activated if there is ever a collision or the system encounters a severe error.

3. System Components

3.1 Vehicle to Vehicle (V2V) Communication

The V2V subsystem is the means for cars to communicate their states with each other. This is essential for determining behavior at intersections.

3.1.1 Protocol Overview

Autonomous vehicles must conform to the same communication protocol if they are operating in the same environment (i.e., the track) in order to ensure safe and predictable behavior, particularly when transiting intersections.

The protocol proposed by Autocross is broken down here into three components: the hardware for sending and receiving signals, a set of broadcast signals corresponding to the vehicle's state, and response behaviors.

The protocol is greatly simplified by the nature of the project environment. Because the track has only one intersection, vehicles can unambiguously transmit their state relative to the intersection without using absolute position data. Thus, it is not necessary for complex image processing or other direct sensory data for vehicles to assert the positions of other vehicles.

3.1.2 Hardware

The communication hardware consists of a single 2.4GHz wireless transceiver

Figure 4: RF24L01 Wireless transmitter and receiver

3.1.3 Broadcast Signals

Vehicles will communicate via basic signals which encode vehicle state and timestamp. As described above, absolute position is not transmitted. V2V communication is solely for handling intersection behavior, and so vehicles only transmit during this phase. Conditions for signal transmission and response behavior is described in the next section.

STOPPED

REQUEST_FOR_TRANSIT

TRANSIT_REQUEST_RESPONSE (int: vehicle_id, bool: accepted)

IN_TRANSIT

TRANSIT_COMPLETED

3.1.4 Behavior

notes : Car emits STOPPED signal when it stops at intersection. It waits for X seconds, listening for STOPPED signals from others vehicles.

Cars listen for any IN_TRANSIT signals. If a car is in the intersection already, wait for TRANSIT_COMPLETED signal.

Cars determine which car should go first, based on timestamp. If there is a conflict, resolve it according to some rule (i.e., lowest vehicle id goes first). Car with priority emits REQUEST_FOR_TRANSIT.

All other cars wait for the REQUEST_FOR_TRANSIT, then emit TRANSIT_REQUEST_RESPONSE with the requesting vehicle's id.

Once priority car has received an affirmative TRANSIT_REQUEST_RESPONSE from all other cars, it commences intersection transit and emits IN_TRANSIT until it is through the intersection.*

Priority car emits TRANSIT_COMPLETED.

Process repeats and the next car is determined.

**Response might be false, indicating some error or deadlock situation that needs to be resolved.*

3.1.5 Variables

Monitored Variables
Inform controller that vehicle is clear to enter intersection

TRANSIT_REQUEST_APPROVED

Controlled Variables

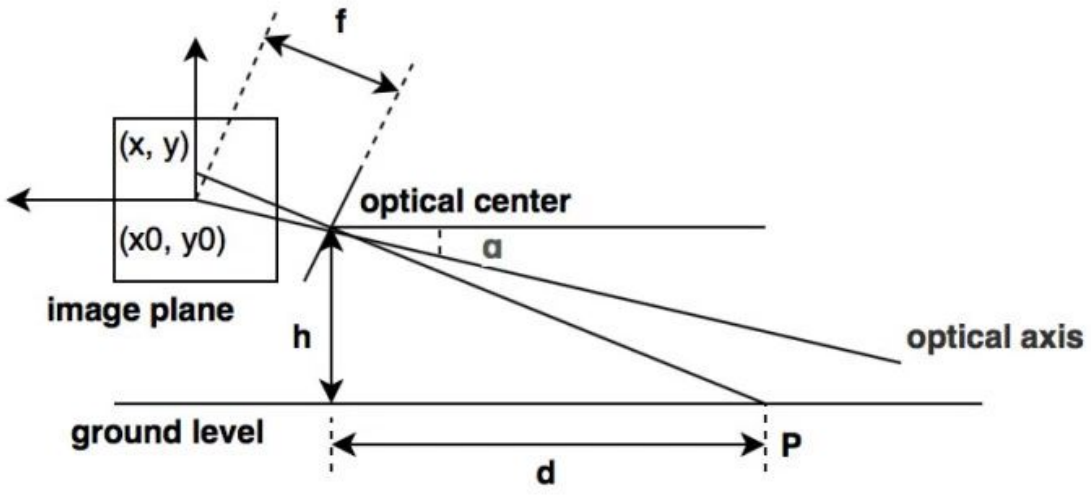
Pass the current actions of the car to the V2V system so it knows what signal to emit. May be booleans or an enumerated state.

STOPPED_AT_INTERSECTION
TRANSITING_INTERSECTION

3.2 Vision Processing

Image processing will serve as the main source of control input for the AutoCross system. The vision system will have the responsibility of detecting lanes allowing for lane following, detecting stop signs, detecting obstacles (in conjunction with ultrasonic sensor), as well as, determining the distance from the camera to objects. The following is a list of responsibilities the vision system has and an explanation of how it will achieve them:

Determine distances to detected objects:
Determine distance between car and detected objects using machine vision. A monocular vision method can be applied to determine distances. This method uses inherent properties of the camera that can be calculated with camera calibration using openCV.



$$d = h/\tan(\partial + \arctan((y - y_0)/f)) \quad (1)$$

$$u = \frac{x}{dx} + u_0 \quad v = \frac{y}{dy} + v_0 \quad (2)$$

Let $x_0 = y_0 = 0$, from (1) and (2):

$$d = h/\tan(\alpha + \arctan((v - v_0)/a_y)), \qquad (a_y = f/dy) \quad (3)$$

Figure 4: Distance detection

Sources:
1: https://zhengludwig.wordpress.com/projects/self-driving-rc-car/
2: http://ieeexplore.ieee.org/document/1336478/authors

Detect Lanes:
Lanes will be detected using canny edge detection. After running canny edge detection lines will be filtered out based on proximity to car (field of view) as well as slope. Once two lanes are determined the angular difference between the medial line of the car and the lanes will be determined. This angular difference will be used in a PID control loop to keep the car on track and finding the following lanes.

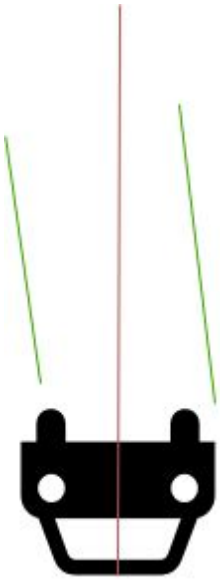


Figure 5: Lane angular offset

Detect and Determined Objects - Stop Sign
Determined objects will be detected using HAAR cascades. HAAR cascades are files that are pre-compiled machine learning files that detect differences between pictures. HAAR cascades files can be trained and created to detect unique objects. In this case, a HAAR cascade file for a

stop sign has been created using a set of positive and negative stop sign pictures. This technique can easily detect the field of view of the object desired which can be interpolated to detect the distance.

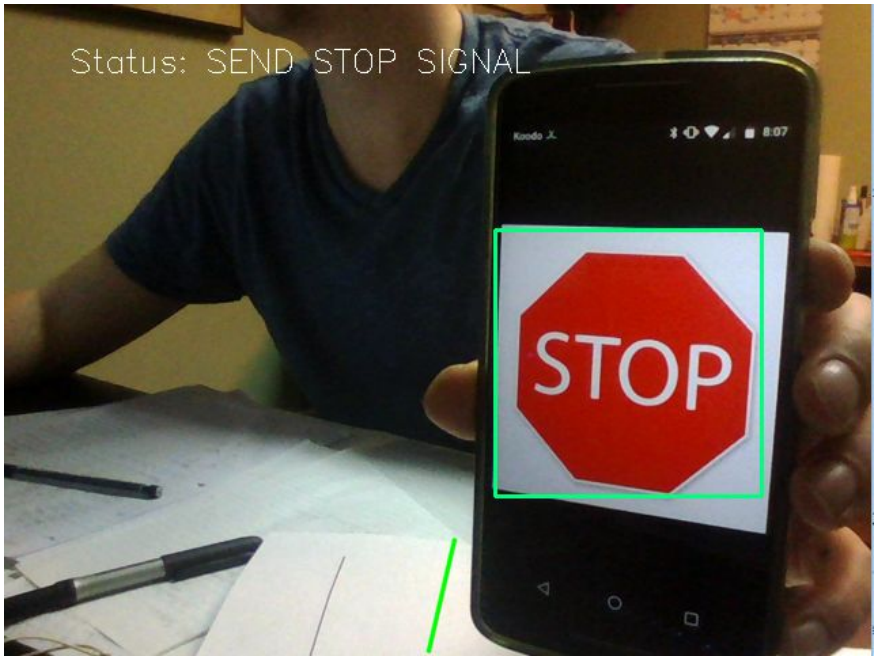


Figure 6: HAAR cascade stop sign detection

Vision Object Detection:
Vision object detection is low priority for this application. The intention is for object vision detection to be used in conjunction with the ultrasonic sensor. Vision object detection will done using blob detection which detects changes in images based on several factors. A detection of many blobs in a local area can detect a path obstruction.

3.2.1 Variables

Monitored Variables

STOP_SIGN_DETECTED
STOP_SIGN_DISTANCE

LEFT_LANE_DIST
RIGHT_LANE_DIST

OBSTACLE_DETECTED
OBSTACLE_DISTANCE

OBSTACLE_POSITION *left/right offset from center of camera*

INSIDE_INTERSECTION *not sure how we are going to do this, but sensors need some way of knowing if we are inside an intersection or not. Maybe distance to stop sign on opposite side? Leaving as a simple boolean for now.*

Controlled Variables
None

3.3 Vehicle Power and Steering (Output)

Hall effect sensor used to monitor speed

3.3.1 Variables

Monitored Variables
ACTUAL_SPEED

Controlled Variables
SET_SPEED *could be negative for reverse - do we assume constant speed? If we can, then we can have +1, -1 , 0 <- discrete speed vars*

I don't so, I think we need to use an actual speed to accurately calculate how long it will take to slow down, accelerate, etc

It would be awesome if we could just say "slow down" and it would come to a stop in a predictable time, but in reality we will probably have to know the actual speed from a hall effect sensor.. Too many things can affect the speed, like avoiding obstacles, low battery, etc

WHEEL_TURN_POSITION
set of discrete values indicating if we want to the wheel to be turned left or right, and by how much. I don't think we need a corresponding monitored variable, since they are just discrete values (not a feedback system) E.g.,
-2: sharp left
-1: slight left
0: straight
1: slight right
2: sharp right

4. Software Architecture

This section specifies the top-level software control and interactions with the vehicle subsystems (see *monitored and controlled variables in System Components).*
James: How everything comes together. The main loop of our Raspberry Pi; how inputs/outputs from subsystems are polled and processed. I think this can be a substitute for MID/MIS. Will work on this tomorrow

Main Control Loop pseudocode

NOTE - the sensor module functions (particularly checkIfInTransit()) are vague. This is going to be one of the hardest parts of the project. It's one thing to follow lanes, it's another to make a right or left turn through an intersection using visual/environmental cues.

```
// function calls to submodules are assumed to read/write global     //
variables (no parameters passed/returned)

Enum STATE = {
      NORMAL_OPERATION,
      STOPPING,
      STOPPED,
      TRANSITING_INTERSECTION,
      AVOIDING_OBSTACLE
}

// randomly choose how to proceed through intersection
Enum ACTION = {
      GO_STRAIGHT,
      TURN_RIGHT,
      TURN_LEFT
}

//constants
VEHICLE_UUID = 12456789ABCDEF
NORMAL_SPEED = 20

// global monitored variables, set by corresponding modules
// power and steering monitored variables
ACTUAL_SPEED = 0

// V2V monitored variables
TRANSIT_REQUEST_APPROVED = null

// image/sensor monitored variables
STOP_SIGN_DETECTED = false
STOP_SIGN_DISTANCE = null
LEFT_LANE_DIST = null
RIGHT_LANE_DIST = null
OBSTACLE_DETECTED = false
OBSTACLE_DISTANCE = null
OBSTACLE_POSITION = null
INSIDE_INTERSECTION = false


// car starts sitting stationary on track
```

```
STATE = NORMAL_OPERATION
ACTION = Random()

ControlModule.setSpeed(NORMAL_SPEED)

While(True){

      switch (STATE){
            Case NORMAL_OPERATION:
                  // get external conditions from sensor module
                  // assumption that obstacles will not be in
                  // intersection (only during normal operation)
                  SensorModule.checkForObstacles()
                  SensorModule.checkForStopSign()
                  SensorModule.getLanePosition()

                  If (OBSTACLE_DETECTED){
                        STATE = AVOIDING_OBSTACLE
                  }Else if (STOP_SIGN_DETECTED){
                        STATE = STOPPING
                  }Else{
                        ControlModule.maintainCourse()
                  }

            Case STOPPING:
                  ControlModule.setSpeed(0)
                  If (ACTUAL_SPEED == 0){
                        STATE = STOPPED
                  }

            Case STOPPED:
                  V2VModule.getTransitApproval()
                  While (not TRANSIT_REQUEST_APPROVED){
                        wait ()
                  }
                  STATE = TRANSITING_INTERSECTION
                  V2VModule.signalInTransit()

            Case TRANSITING_INTERSECTION:
                  While (true){
                        ControlModule.action() // go left, right, or straight
                        SensorModule.checkIfInsideIntersection()
                        If (not INSIDE_INTERSECTION){
                              V2VModule.signalTransitComplete()
                              STATE = NORMAL_OPERATION
                              ACTION = Random() //next intersection
                              break
                        }
```

```
                }

            Case AVOIDING_OBSTACLE:
                // complicated subroutines go here. ¯\_(ツ)_/¯
        }

}
```

5. MIS and MID

5.1.1 MIS

ControlModule - Main function that controls the motion and reaction of the car to an y obstacles.
Type - Enum
Uses - None

5.1.2 MID

setSpeed - Method of function ControlModule that sets a speed for the RC car
//Accepts Argument of type - Float?
Type - Float
Uses - ControlModule

5.1.3 MID

maintainCourse() - Method of the function ControlModule
Type - Float
//Uses - OBSTACLE_DETECTED()
Uses - ControlModule

5.1.4 MID

action() - Method of the ControlModule used to specify what direction car needs to move in.
Type - Binary
Uses - ControlModule

5.2.1 MIS

SensorModule - Function to control input/output of the different sensors.
Type - Enum
Uses - None

5.2.2 MID

checkForObstacles() - Method of function SensorModule that polls for any obstacles on the road.
Type - Binary
Uses - SensorModule

5.2.3 MID

checkForStopSign() - Method of the function SensorModule that polls the Supersonic sensor to check for stop signs on the road.
Type - Binary
Uses - SensorModule

5.2.4 MID

getLanePosition() - Method of the function SensorModule that polls the Hall Effect Speed Encoder and Pi Camera to give current lane position.
Type - Binary
Uses - SensorModule

5.2.5 MID

checkIfInsideIntersection() - Method of the SensorModule used to check if car is inside the intersection.
Type - Binary
Uses - SensorModule

5.3.1 MIS

V2VModule - Function that implements the Vehicle-to-Vehicle communication protocol and associated algorithms.
Type - Enum
Uses - None

5.3.2 MID

getTransitApproval() - Method that polls for right of passing the intersection
Type - Binary
Uses - V2VModule

5.3.3 MID

signalInTransit() - Method of the function V2VModule used to declare state of the car at an intersection.
Type - Binary
Uses - V2VModule

5.3.4 MID

signalTransitComplete() - Method of the V2VModule used to state if car has exited intersection.
Type - Binary
Uses - V2VModule